Sheet (1, 2, 3, 4, 5, 6, 12, 13, 19, 20, 28)

Input/Output Organization

(4.1)

The input status bit in an interface circuit is cleared as soon as the input buffer is read? Why?

Solution: →

After reading ~~for~~ the input data, it is necessary to clear the input status flag $SIN = \boxed{0}$ before the program begins a new read operation.

(4.2)

address Bus = 16 bit = A_{15-0}

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
						X	X								

A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

(8 4 2 1)

$7CA4_H$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	0	0	1	0	1	0	0	1	0	0

XX

$7CA4_H$ =	0111	11	00	1010	0100	Device 1
$7DA4_H$ =	0111	11	01	1010	0100	Device 2
$7EA4_H$ =	0111	11	10	1010	0100	Device 3
$7FA4_H$ =	0111	11	11	1010	0100	Device 4

4.4

Subroutine	Interrupt Service Routine (ISR)
A Subroutine is called by a program instruction to perform a function needed by the calling program. يتم مناداته داخل أي برنامج وذلك بهدف أداء وظيفة معينة للبرنامج الرئيسي ~~عند~~ مناداته.	• ISR is intiated by an event such as input operation or hardware error. • The function it performes may not be at related to the program being executed at the time of interruption. • It must not affect any of the data or status information relating to that program. ISR يتم تنفيذه عند حدوث أي عملية إدخال أو حدوث أي مشاكل في (Hardware) ISR : الوظيفة التي تقوم بها ليس لها علاقة بالبرنامج الرئيسي الذي تم مقاطعته أثناء وجوده في الـ(processor) ISR : لا تؤثر على أي (data) أو على (Status) بالبرنامج الرئيسي.

4.5 من هذا الفصل اتفقنا على أن ال processor لا يستجيب الى (Interrupt) إلا بعد أن يقوم بتنفيذ الأمر الذي يقوم بتنفيذه

ماذا لو حدث واستجاب (processor) إلى (Interrupt) في منتصف تنفيذ أمر معين؟

Discuss Diffculties that may arise.

If execution of the interrupted instructions is to be completed after return from interrupt, a large amount of information needs to be saved. This includes the contents of any temporary registers, intermediate results, the address of the next instruction, ... etc

An alternative is to abort the interrupted instruction and start its execution from the begining after return from interrupt.

In this case ⇒

the results of an instruction must not be stored in register or memory location until it guaranteed that execution of the instruction will be completed without interruption.

إذا جاء Interrupt إلى ال processor أثناء تنفيذ أمر معين وهناك حالتين ① يتم مقاطعة processor بعد استكمال الأمر الذي ينفذه ففي هذه الحالة لابد من تخزين كمية كبيرة من البيانات مثل مسجلات (Registers) والنتائج وعنوان الأمر التالي من التنفيذ.

الحل الآخر ⇐ ② هو عند مقاطعة ال(processor) الأمر الذي تم مقاطعته يتم إعادة مرة أخرى بعد الرجوع من المقاطعة وهنا لا نحتاج إلى تخزين أي شيء.

(4.6)

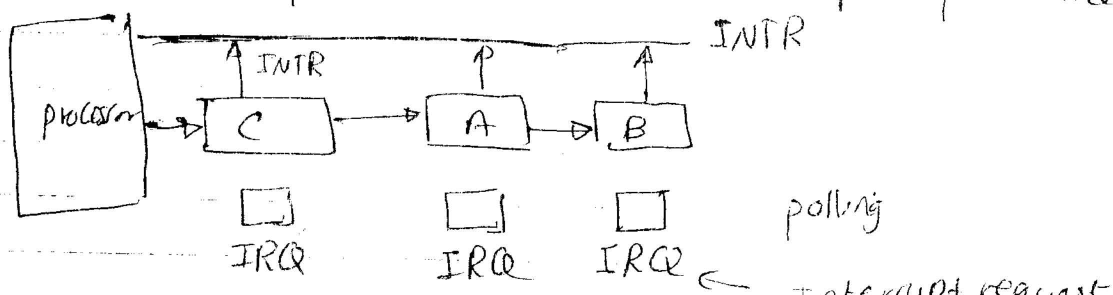
problem: Three devices A, B, C

- Interrupt nesting for devices A an B is not allowed,
- But interrupt requests from C may be accepted while either A or B is being serviced.

Suggest different ways in which this can be accomplished in each of the following.

(a) the computer has one interrupt-request line

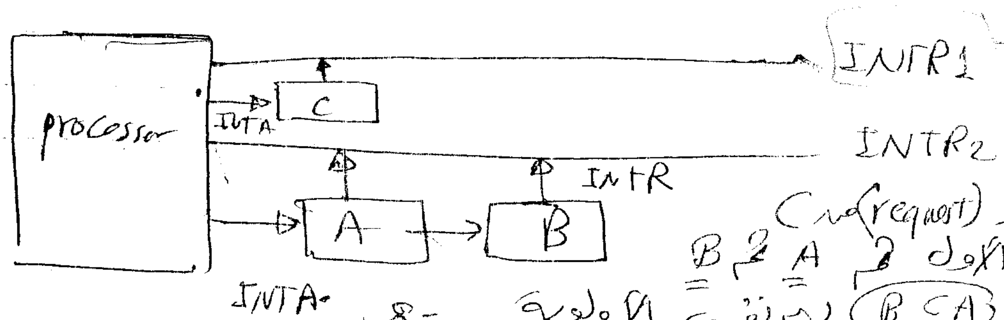
Solution



IRQ 1 الجهاز الذي يطلب (Interrupt) نقوم بوضع
وعندما يمر الـ (Processor) على الجهاز بالترتيب الذي يحدده
IRQ=1 نقوم بخدمته وتنفيذ (ISR) الخاصة به

(b) we have 2 lines $INTR_1$, $INTR_2$
and $INTR_1 \overset{\text{(له أولوية)}}{>} INTR_2$.

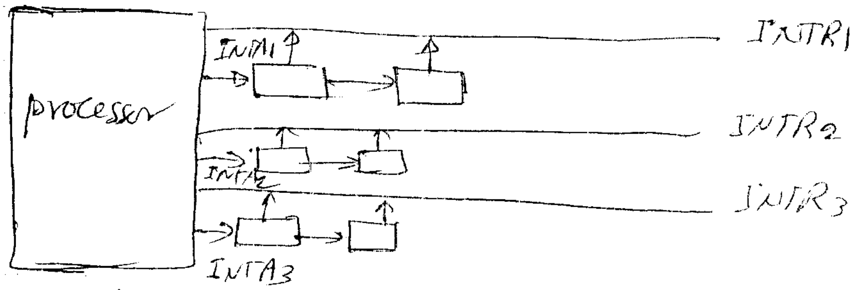
Solution



إذا طلب (request) C نخدم الأول ثم A ثم B
وكانت (B C A) لهم نفس الأولوية

(4.12)

Design a Logic circuit to implement the priority shown in figure 4.8 b



اثر وصول

INTR i $\xrightarrow{\text{الاشارة تخرج}}$ INTA i

Interrupt request   interrupt acknowledge

Note

(periority INTR$_1$ > INTR$_2$ > INTR$_3$)

(Req)

ⓐ give a truth table for each of output

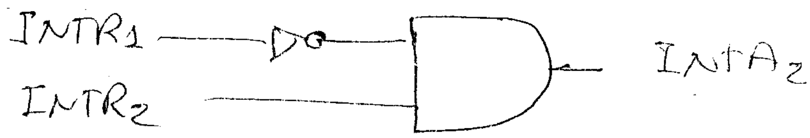
equations →
$$\begin{cases} INTA_1 = INTR_1 \\ INTA_2 = INTR_2 \cdot \overline{INTR_1} = 1\;0 \quad 0\;1 \\ INTA_3 = INTR_3 \cdot \overline{INTR_1} \cdot \overline{INTR_2} \quad 0\;0\;1 \end{cases}$$

INTR1	INTR2	INTR3	INTA1	INTA2	INTA3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	1	0	0

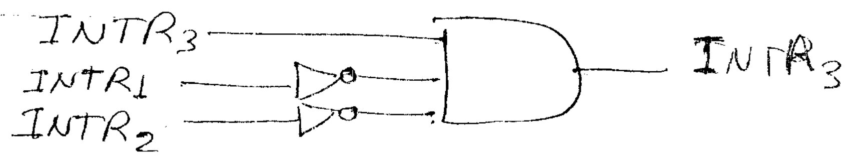
(b) Logic circuits

- $INTA_1 = INTR_1$

- $INTA_2 = INTR_2 \cdot \overline{INTR_1}$



- $INTA_3 = INTR_3 \cdot \overline{INTR_1} \cdot \overline{INTR_2}$



(c) Yes, My design extended easily for more interrupt request Lines.

(d) $\boxed{\text{Add Inputs}}$:-

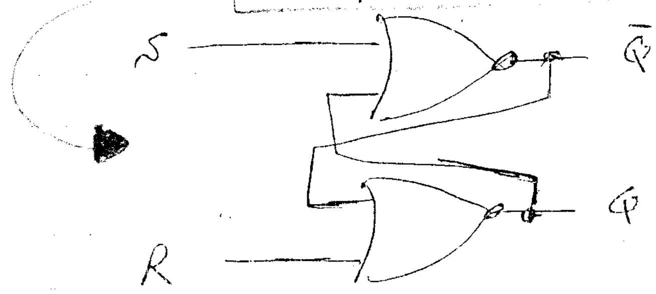
modify circuit

when Decide : 1 @↑ → $INTA_i = 1$

Reset : 0 @↑ ⇒ $INTA_i = 0$

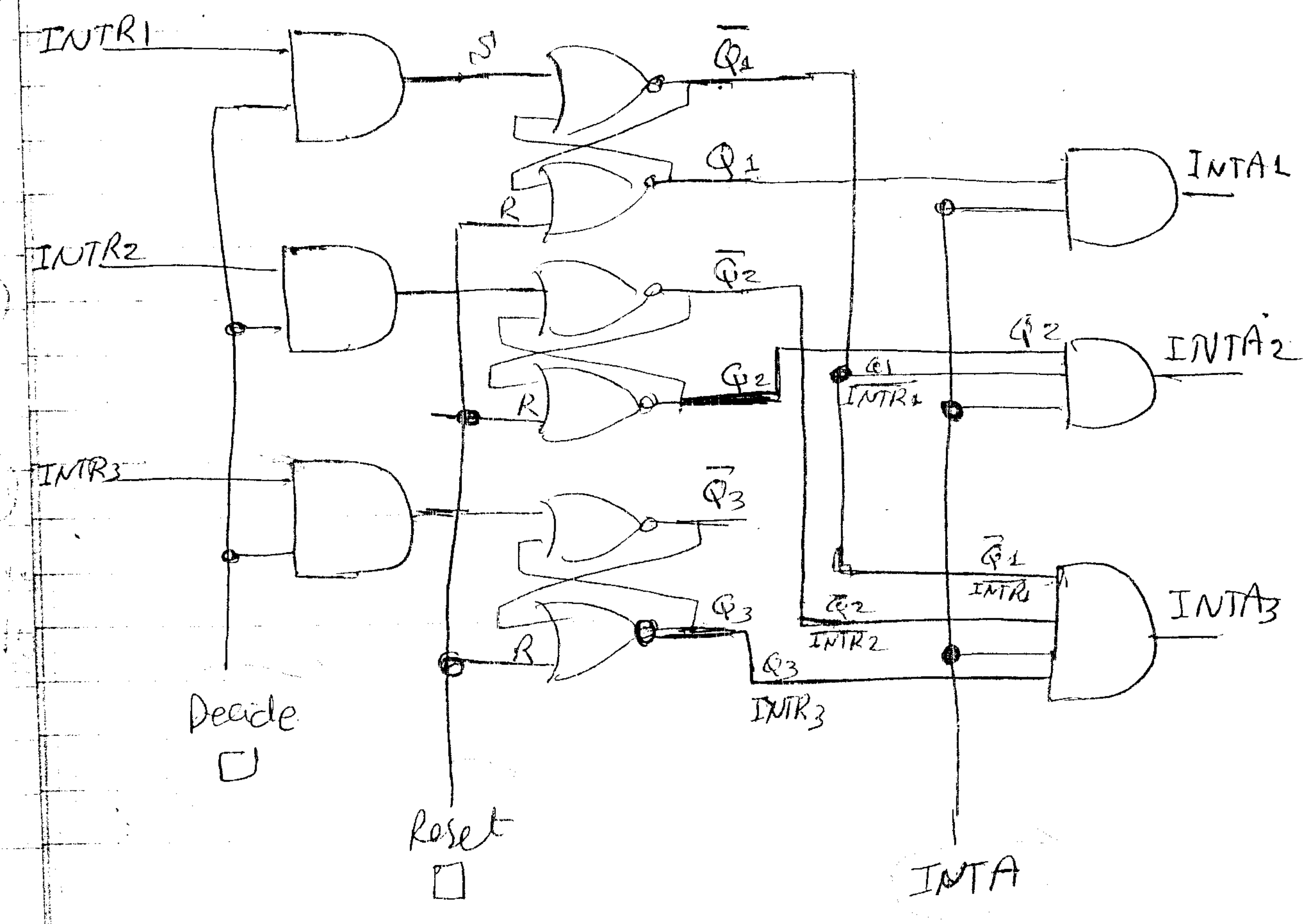
Design :-

Note :-> S-R flip flop



S	R	Q
0	0	No change
0	1	Reset Q=0
1	0	Set Q=1
1	1	Not allowed.

$$INTA_1 = INTR_1 \cdot Decide$$
$$INTA_2 = INTR_2 \cdot \overline{INTR_1} \cdot Decide$$
$$INTA_3 = INTR_3 \cdot \overline{INTR_1} \cdot \overline{INTR_2} \cdot Decide$$



4-13

(هذه الصفحة بها المسألة فقط)

Design a circuit for rotating periority for selecting one of several requests based on their periority.

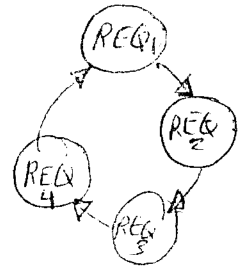
Input Lines (Request Lines)

REQ_1 , REQ_2, REQ_3, REQ_4

$\boxed{\text{periority of}}$: $REQ_1 > REQ_2 > REQ_3 > REQ_4$

بعد أن يأخذ أي Line منهم الخدمة يصبح هو الأقل في الأولوية: فمثلاً إذا أخذ (REQ_2) الخدمة يصبح هو الأقل ويصبح ترتيب الأولويات

$REQ_3 > REQ_4 > REQ_1 > REQ_2$



Design circuit should include

four Grant signals ⇒ GR1 through GR4

واحدة فقط منهم هي اللي تصبح بواحد إذا جاءت pulse على دخل يسمى (Decide)

Whole circuit

$\boxed{\text{Inputs}}$: REQ_1 , REQ_2, REQ_3, REQ_4
Decide

$\boxed{\text{output}}$ GR_1, GR_2, GR_3, GR_4.

Solution

ملاحظات

هذه الصفحة بها
المقترحات التي يستخدم
في حل المسألة

Register A

① تقوم بتخزين ومعرفة أي Device قد أخذ Grant آخر واحد.

② واحدة فقط bit output A هي التي تساوي واحد عند أي لحظة زمنية.

Arbitration Request ← طلب أخذ Bus
REQ1, REQ2, REQ3, REQ4

arbitration Grant ← معناها أي Device قد أخذ Bus بالفعل .. GR1, GR2, GR3, GR4

Decide → falling edge

Decide (falling edge) ← تقوم بعمل Record لـ (current arbitration cycle) على Register A ← أي أنه يسجل ((Grant Device))

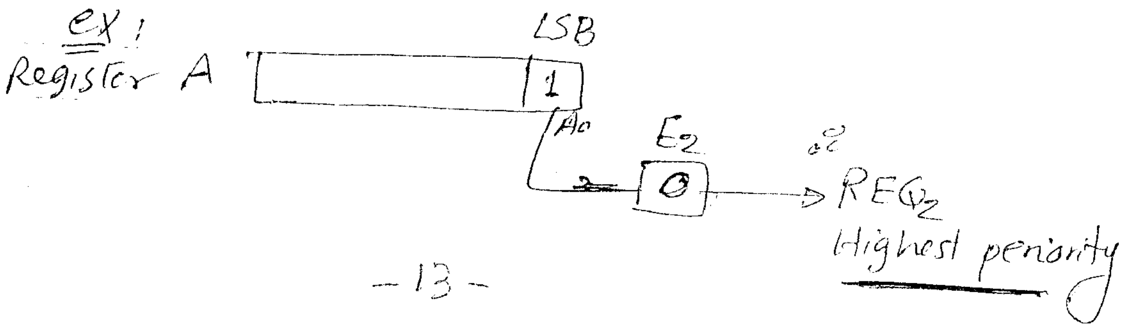
وفي نفس الوقت سوف يأتي على Register B Record (يسجل) أي (new Request)

(هذا لكي نمنع أي أحد (Request) جديد سوف يؤثر على (Grant)

Circuit intialization

- one bit of Register A is equal to one Which will be the output to the highest Periority Line.

ex:



How the Circuit works :-

$\boxed{REQ_1 = 1}$

assume that REQ_1 takes the bus

$GR_1 = 1$, $E_1 = 0$

$\therefore GR_1 = 1 \quad \therefore A_1 = \overline{GR_1} = 0$

$$E_2 = \overline{\overline{A_1}\cdot(\overline{E_1} + REQ_1)} \quad \Rightarrow E_2 = 1\cdot(0+1) = 1$$

$$GR_2 = \overline{(\overline{E_2}\cdot REQ_2)} \qquad GR_2 = \overline{(0\cdot 1)} = 1$$

2- equations ركز

$$E_{i+1} = \overline{A_i}\cdot(E_i + REQ_i)$$

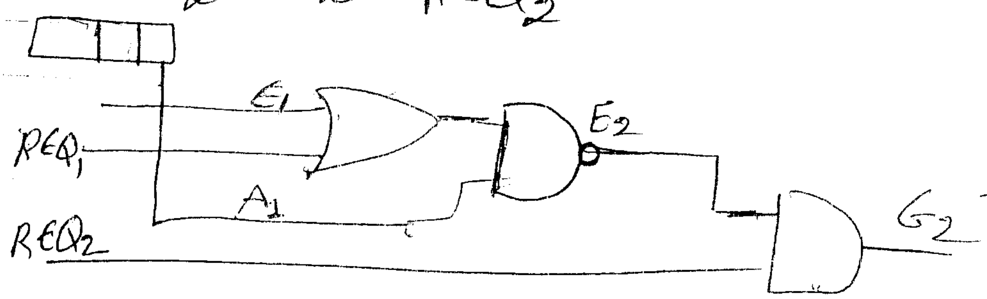
$$i = i+1 \% 4$$

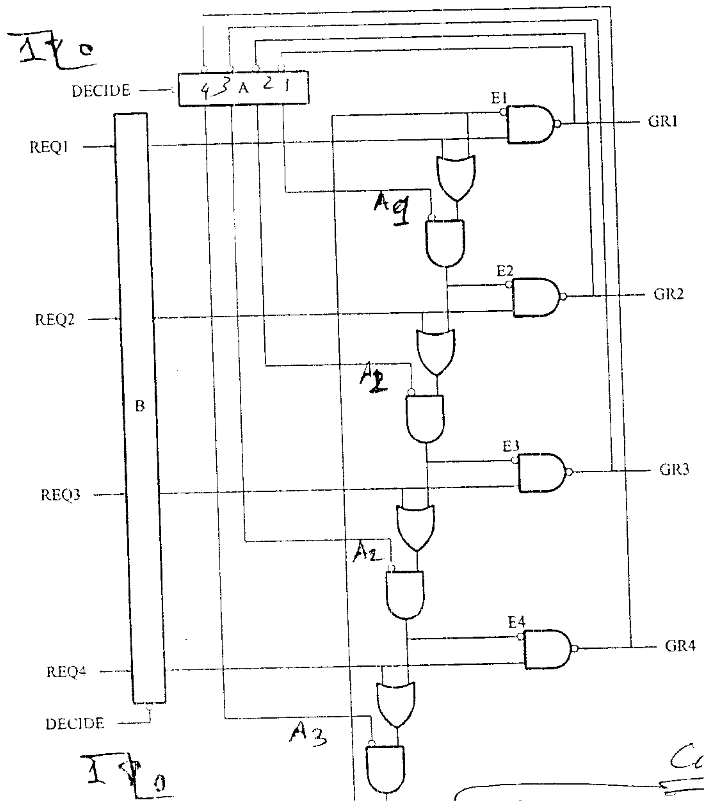
$$GR_{i+1} = \overline{\overline{E}_{i+1}\cdot REQ_{i+1}}$$

$\boxed{\text{assume}}$ $i = 1$

$$E_2 = \bar{A}_1\,(E_1 + REQ_1)$$

$$GR_2 = E_2 \cdot REQ_2$$





Care !-

intial $\boxed{REQ_1 = 1,\ E_1 = 0,\ GR_1 = 1}$

so $A_1 = \overline{GR_1} = 0$

4.14. The truth table for a priority encoder is given below.

1	2	3	4	5	6	7	IPL_2	IPL_1	IPL_0
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1
x	1	0	0	0	0	0	0	1	0
x	x	1	0	0	0	0	0	1	1
x	x	x	1	0	0	0	1	0	0
x	x	x	x	1	0	0	1	0	1
x	x	x	x	x	1	0	1	1	0
x	x	x	x	x	x	1	1	1	1

$\Rightarrow E_1 + REQ_1 = 1$

$\therefore E_2 = \left[\overline{A_0} \cdot (E_1 + REQ_1)\right]$

$E_2 = [1 \cdot 1] = 1$

if $REQ_1 = 0 \Rightarrow E_2 = 0$

$\boxed{\text{assuma } REQ_2 = 1}$

A possible implementation for this priority circuit is as follows:

$\therefore GR_2 = \overline{E_2} \cdot REQ_2$

$= 1 \cdot 1 = 1$

if $GR_2 = 1$ & $REQ_2 = 0$.

$\Rightarrow \therefore A_2 = 1$, $\overline{A_2} = 0$

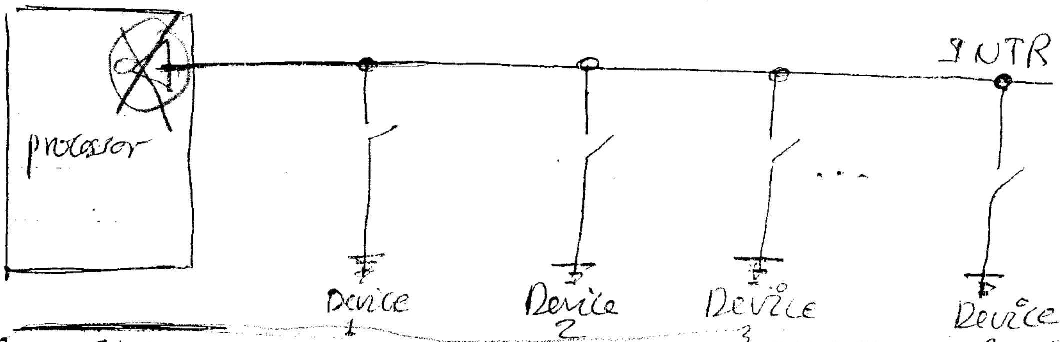
4.19

Interrupt request Line ⇒ Use open collector Scheme,
Carries a signal that is the logical OR of the requests from all the devices connected to it.

Req:-

is req. to Generate a Signal that indicates that all devices connected to the bus are ready.

$\boxed{\text{Solution}}$



$$INTR = (INTR + INTR_2 + INTR_3 + \cdots INTR_n)$$

- Each device pull the line down (closes a switch to a ground) when it is not ready
- It open the switches when it is ready
- Thus, **the Line will be high** when all devices are ready.

$\boxed{\text{device}}$ لما تكون

Switch

not ready

Ground تعمل توصيل الـ

وهنا تعمل ارت

Interrupt Request (INTR) = 0

Ready

(Switch) ~~توصيل~~ تقوم بفتح

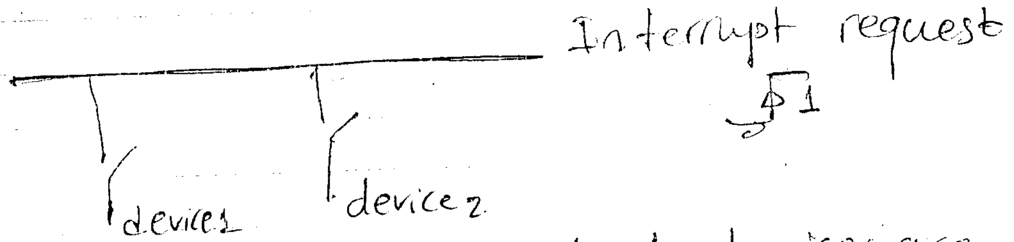
وهنا تجعله

switch open

∴ تكون INTR = 1

لأنه غير واصل بالأرث

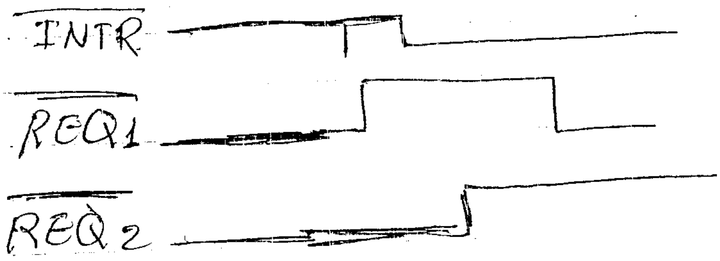
4.20



what happen if two independent devices are Connected to this Line?

Solution

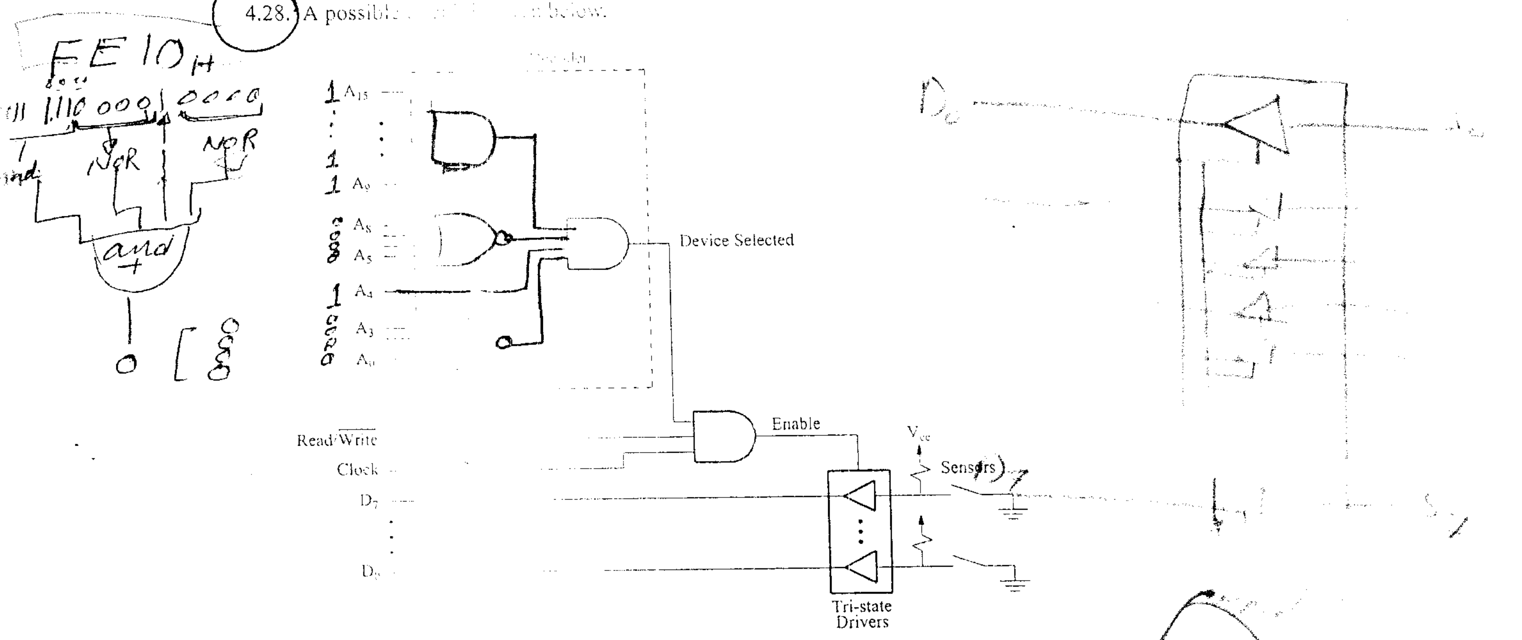
The request from one device may be masked by other because the processor may see only one device Request



4.27. A larger distance means longer delay for the signals traveling between the processor and the input device. Primarily, this means that $t_2 - t_1$, $t_3 - t_2$ and $t_5 - t_3$ will increase. Since longer distances may also mean larger skew, the intervals $t_1 - t_0$ and $t_4 - t_5$ may have to be increased to cover worst-case differences in propagation delay.

In the case of Figure 4.24, the clock period must be increased to accommodate the maximum propagation delay.

4.28. A possible circuit is given below.



عشان نقدر نقرأ القراءة من ال Sensors لازم من أن نتحقق الآتي
((Tri-state Buffer)) ال Control وهو $\boxed{Enable = 1}$

$D_7 - D_0$ lines ← sensor readings
enable = 1

faling edge clock

تكون Enable = 1 لـ ① إذا كان هو العنوان
② حينما يكون (Read) = 1
③ حينما نكون الجهاز (device) الذي سنقرأ منه.

address
Read
Clock
Enable

FE10

Clock

11

— 18 —